Sistema de gestión avanzada de datos basado en Flujos de Trabajo y análisis de lenguajes estructurados

Víctor Bravo
Universidad de los Andes ULA
Facultad de Ingeniería
Mérida, Venezuela
Email: victorrbravo@gmail.com

Solazver Solé
Universidad de los Andes ULA
Facultad de Ingeniería
Email: solazver@ula.ve

***Abstract*—En este trabajo se describe una propuesta para la construcción rápida y robusta de sistemas de información que contengan lógicas complejas y que puedan ser desplegados en distintos ambientes y plataformas computacionales. Para ello, se hace uso de los conceptos fundamentales de base de datos relacionales, redes de petri y flujos de trabajo, así como también se toma como referencia los lenguajes de procesos de negocio, tales como BPM y BPEL. Con la finalidad de ofrecer una visión modular se discute sobre los diferentes componentes que conforman el sistema SAFET, elemento central del trabajo, subrayando los procesos de coordinación e integración entre ellos. Finalmente, se muestran dos ejemplos de uso común en internet, donde se pueden identificar de forma práctica algunos aportes de la propuesta al desarrollo de sistemas de información.**

***Index Terms*—Flujo de Trabajo, SQL, BPM, XML, JSON, Red de Petri**

1. Introducción

Cualquier organización tiene definido procesos que agrupan sus actividades diarias, entre las cuales generalmente se incluyen el manejo y procesamiento de datos y el resumen de información para la toma de decisiones. Las actividades cotidianas requieren tiempo y están supeditadas a condiciones específicas para su correcta ejecución. En relación a ello, las tecnologías de la información generalmente sirven como mecanismos para la mejora de procesos.

Este trabajo muestra la estructura y funcionalidades de una herramienta que responde a las situaciones planteadas anteriormente. Uno de los propósitos principales, es presentar un sistema computacional que flexibilice la comunicación entre personas y sistemas de información a través del uso de un lenguaje específico para flujos de trabajo, utilizado para la construcción de vistas inteligibles, ordenadas y adaptadas de datos. Es importante señalar que en este trabajo se propone una visión específica para cada uno los elementos fundamentales que conforman un sistema de información, que van desde el acceso y permisos para los activos de información hasta la visualización de datos.

En este sentido, se utilizan los lenguajes de marcado tales como JSON [3] y XML [7] para la definición de esquemas de la entrada y salida de datos que permitan un rápido desarrollo de aplicaciones, incorporando de esta forma características de seguridad, integración y gestión de recursos.

2. Base Teóricas

Con el objetivo de contextualizar la propuesta se describen brevemente algunos conceptos utilizados en el diseño e implementación del sistema, tales como la idea de flujos de trabajo basado en Redes de Petri y las bases de datos relacionales Es imprescindible en una primera aproximación: discutir las nociones teóricas que acompañan al proceso de definición del sistema.

2.1. Redes de Petri

Las ideas sobre Redes de Petri [11] se iniciaron en los años 60, y se planteaban como una herramienta para resolver problemas en el área de los sistemas distribuidos. Una Red de Petri es un lenguaje para modelado matemático compuesto por elementos tales como transiciones, nodos, arcos dirigidos, y fichas o marcas ubicadas en cada nodo [11]. La configuración de las marcas o fichas pueden variar según eventos, estados o tiempo. Formalmente se puede describir una Red de Petri como una tupla (S,T,F,m0) constituida por nodos, transiciones, movimientos y una condición inicial respectivamente.

Se han realizado diferentes trabajos que extienden la definición formal de una Red de Petri. Una de estas extensiones es la Red de Petri Coloreada [1] (Coloured Petri Net en inglés), que agrega el hecho de poder identificar a cada ficha por un determinado color, convirtiendo las fichas en elementos distinguibles unos de otros. Otra de las extensiones recientes es el propuesto por Hofstede et al. en [3], donde se describe el lenguaje denominado YAWL, que entre otras características incorpora el uso de los operadores XOR, AND, OR de entrada (JOIN) y salida (SPLIT) para modelar la concurrencia y paralelismo entre eventos o actividades.

2.2. Base de datos Relacionales

Las Bases de Datos Relacionales (BDR) son un fundamento en la construcción de sistemas de información modernos y profesionales. Este hecho se puede atribuir entre otras causas, a la disponibilidad de un Lenguaje de Consulta estructurado llamado SQL (Structured Query Language en inglés), que prácticamente se ha convertido en un estándar para aplicaciones de gestión de datos (ver [4] y [5]). Por otro lado el desarrollo a través de muchos años de algoritmos para mejorar la eficiencia computacional de los denominados gestores o servidores de bases de datos relacionales, es otro argumento que apoya este hecho. Las BDR se basan en un modelo relacional de los datos donde una base de datos se puede ver como una colección de relaciones, donde cada relación está conformada por un conjunto de tuplas que tienen una cantidad de atributos finita. Es importante señalar, que por definición todos los elementos de un conjunto son distintos y como una relación se define como un conjunto de tuplas entonces todas las tuplas en una relación deben ser distintas. Por esta razón generalmente en cada relación se establece al menos un atributo como clave principal, cuya función es identificar unívocamente a cada tupla y evitar la duplicidad de datos, además de servir como elemento de base para algoritmos de verificación y simplificación de las relaciones. El modelo relacional de datos contempla que ningún atributo establecido como clave principal puede tener valores nulos, puesto que esto no permitiría la identificación de esa tupla.

Uno de los lenguajes disponibles para el modelo relacional es el álgebra relacional, que es un lenguaje formal procedimental útil para realizar algunas operaciones sobre los datos de una BDR. Por otro lado, se dispone del SQL como lenguaje práctico para las BDRs, que en cierta forma está basado en los conceptos del álgebra y el cálculo relacional. Las operaciones básicas o primitivas para el álgebra relacional son: Selección, Proyección y Reunión natural (también llamada concatenación). La selección permite obtener un subconjunto de tuplas de la relación evaluada que cumple con una condición determinada, en SQL se puede utilizar la cláusula WHERE para realizar una selección. La proyección sirve para buscar un subconjunto de atributos de una relación, su equivalente en SQL se puede obtener usando la cláusula SELECT. La concatenación entre dos relaciones R y S, genera una relación formada por tuplas donde los atributos comunes de R y S coinciden en los valores, las tuplas resultantes incluyen la unión de todos los atributos de R y S, en SQL se puede utilizar la cláusula JOIN para este propósito. Otras operaciones de álgebra relacional son el Producto Cartesiano, Diferencia, Unión e Intersección (heredadas de la teoría de conjuntos) (ver [6])

2.3. Lenguaje de flujos de trabajo

Uno de los elementos principales de esta propuesta es el modelado de los problemas de gestión de información utilizando flujos de trabajo. Por ello se debe considerar un enfoque similar al de los lenguajes para el modelado de procesos de negocio (BPM) [1]. Este enfoque lleva a definir mediante un conjunto de abstracciones procesos de negocio que pueden traducirse de forma automática en programas para la gestión avanzada de datos.

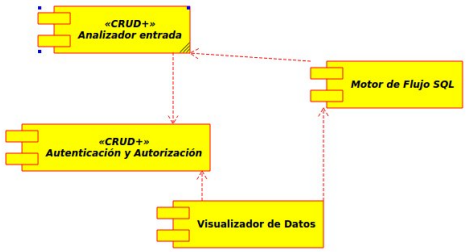

Figure 1. Modelo de componentes de SAFET

Esta tecnología ha servido como solución al modelado rápido de procesos de negocios, donde lenguajes como BPEL [5], extraen elementos importantes para los procesos de negocios como las actividades, tareas y condiciones, a éstas últimas se les pueden agregar los eventos de aceptación. Dentro de la gama de aplicaciones, es preciso contar herramientas que permitan a usuarios y administradores realizar las tareas usuales en el manejo de documentos: la visualización del documento, la visualización de los datos vinculados, así como su segura verificación y validación.

3. Propuesta

El Sistema Avanzado de Gestión de Datos (SAFET) [12] está conformado por el gestor de autenticación y autorización, un analizador de datos (entrada), el motor de flujo de trabajo y el visualizador de datos. tal como se observa en la Figura 1. Todos estos componentes permiten la construcción de sistemas de información de forma rápida y organizada, además de integrar en ellos la validación de datos conforme a los estándares de datos basados en los lenguajes XML y JSON.

En esta sección se muestra la estructura de SAFET y los esquemas de coordinación entre sus componentes. Describiendo así sus elementos: el analizador de lenguaje para especificación de flujo de trabajo, la conexión ubicua a bases de datos relacionales vinculadas a las fichas (tokens), el motor de cálculo de documentos, el visualizador de grafos. Los componentes de SAFET funcionan bajo un modelo CRUD [2] extendido que se describe en la siguiente sección, y que tiene como elemento central la definición de operaciones sobre datos con una semántica particular.

3.1. El modelo CRUD+

Antes de describir los componentes de SAFET, es necesario indicar que se basa en un Modelo CRUD extendido. El modelo CRUD (*Create, Read, Update and Delete*, por sus siglas en inglés) propone modelar la interacción con datos utilizando cuatro operaciones básicas :Creación de un registro, Lectura de un registro, Actualización de un

Table 1. ATRIBUTOS DEL USUARIO

Atributos	
auth.account.1	Nombre del usuario
auth.pass.1	Contraseña cifrada
auth.realname.1	Nombres del usuario
auth.role.1	Roles o grupos del usuario
auth.tickets.1	*Tokens* para servicios

Table 2. ATRIBUTOS DE LAS OPERACIONES

Atributos	
permises.operation.1	Nombre de la operación
permises.accounts.1	Usuarios que pueden realizar la operación
permises.types.1	Tipo de acciones
permises.roles.1	Lista de roles que pueden realizar la operación

Registro y Eliminación del Registro. SAFET agrega a este esquema la operación "+" que significa el cambio de estado de un registro o elemento (*item*) vinculando un campo de la relación al estado que cambiará. De esta misma forma, se utiliza otra relación (tabla) para llevar el registro de los datos del cambio (evento) ejecutado. Estos datos son la fecha y hora del cambio, el usuario vinculado y otros datos adicionales.

La operación de cambio de estado, resulta una operación común en los sistemas informáticos, y puede ser equiparable a las otras acciones del CRUD. Se podría citar como ejemplos de la operación "+" el cambio de una tarea en un proyecto, la compra de un artículo en un sistema de comercio electrónico o la evaluación de un estudiante en un curso.

Seguidamente se describen los componentes de SAFET y se relacionan con el patrón CRUD extendido, de tal manera de mostrar las ideas fundamentales de la propuesta.

3.2. Gestor de Autenticación y Autorización

SAFET posee un sistema de autenticación y autorización independiente basado en operaciones del modelo CRUD+. Las acciones en SAFET modelan casos de uso basados en operaciones de adición, lectura, modificación, eliminación y cambio de estado. Exite una lista de usuarios, que poseen los campos que se listan en la tabla 1.

Por otro lado se deben configurar los elementos "Role" (Rol) y los elementos "Permises" (Permisos) del sistema. En las lista de roles, grupos o usuarios se usa el caracter ";" para separar los nombres. Estos últimos se asocian con las acciones del modelo CRUD+. El elemento "Permise" contiene los campos que se especifican en la tabla 2.

El elemento "Role" contiene los campos que se especifican en la tabla 3.

La configuración de todos estos elementos permiten especificar un sistema de Autorización y Autenticación flexible, basado en el modelo estándar RBAC [9], y que permite de forma rápida y organizada establecer el acceso organizado a los activos de información.

Table 3. ATRIBUTOS DE LOS ROLES

Campos	
roles.name.1	Nombre del Rol o Grupo
roles.description.1	Descripción del Rol

3.3. Analizador de datos

Uno de los elementos principales del sistema propuesto es el lenguaje de descripción basado en flujos de trabajo. Este lenguaje permite especificar la lógica de los estados, realizar búsquedas avanzadas y visualizar conjuntos de datos (Ver figura 1). Para su implementación se hace uso de la tecnología XML/DTD [7].

Un archivo en formato DTD determina una sintaxis a seguir para un conjunto de archivos XML que es equivalente a la definición de una gramática o autómata de pila. El código 1 contiene algunas etiquetas que pueden estar presentes y se utilizan para construir en memoria una estructura de datos que representa una Red de Petri Coloreada, que posteriormente puede ser dibujada por el componente graphviz como un grafo. La especificación completa de las etiquetas del lenguaje de descripción de flujo de trabajo puede verse en [17]. En el trozo de documento que se muestra en el código 1 se tienen las etiquetas: workflow que agrupa todos los demás elementos; la etiqueta task que especifica lugares o actividades; Variable se utiliza para mostrar una lista de documentos relacionado con la tarea o actividad y el atributo query de la etiqueta Port se utiliza para especificar el evento o restricción de salida/entrada entre tareas utilizando sentencias del lenguaje SQL. Esta especificación indica la forma en que se realiza y verifica una o varias características sobre un documento electrónico

3.4. Motor de cálculo de Flujo de trabajos basado en SQL

El motor de cálculo de flujos de trabajo se implementa a través de una librería dinámica, esto es, una pieza de software que incluye todos los objetos y funciones basadas en el esquema propuesto necesarias para la ejecución de consultas al sistema de información (Ver Figura 1). Este motor realiza evaluaciones en serie o de forma paralela de las condiciones de entrada y salida especificadas en un determinado documento de flujo de trabajo a utilizar compatible con la especificación XML/DTD descrita en [17]. Para ello el motor va evaluando secuencialmente las consultas escritas en lenguaje SQL estándar especificadas en los atributos query de los elementos Port (ver Código 1). El orden de la evaluación se realiza desde la condición inicial hasta la condición final. En las actividades de cualquier organización es frecuente que muchas situaciones se parezcan y hacen que los objetos de información se comporten de forma similar. Para este caso es aplicable el uso de patrones de comportamiento. SAFET utiliza una lista de patrones básicos que engloban el modelado de una gran cantidad de casos reales. Los patrones actúan en las transiciones y son de dos

tipos: salida (SPLIT) y entrada (JOIN). Cada uno tienen tres tipos de comportamiento expresados a través de operadores: Y Lógico (AND), O Lógico (OR) y O Lógico Exclusivo (XOR). Para la utilización de un patrón es necesario que un puerto (Port) tenga conexiones (connection) a dos o más tareas, ya que son operadores binarios. Por ejemplo, si se necesita obtener un listado de todas las solicitudes generadas por Director Ejecutivo o del Director Técnico de una determinada organización. Es posible definir una tarea (Task) que utilice un patrón de conjunción (JOIN), y el operador lógico "o" (OR). El objeto Variable de la nueva tarea contendrá la suma de las solicitudes que han sido registradas por al menos uno de los dos directores.

```
<task id="<id_tarea>"
title="" report="<yes/no>" >
 <port side="forward" type="split" >
 <connection source="<tarea_destino>"
 query="<Consulta_SQL>"
 options="<Opción_SQL>"  >
 </connection>
 </port>
 <variable
 id="<id_variable>"
 scope="task"  type="sql" config="1"
 tokenlink=""
 documentsource="<Campos_SQL>" >
  </variable>

       </task>
```

Código 1. Definición de una tarea del flujo de trabajo

Por otro lado, el motor de cálculo también incluye un módulo para el procesamiento de formularios que puedan ser utilizados en aplicaciones de escritorio, mòvil o o Web. De esta manera las acciones de adición, consulta, modificación y eliminación de datos de los repositorios de información pueden ser definidos utilizando un lenguaje basado en XML/DTD y cuya especificación completa se encuentra en [18].

Debido a que el motor de cálculo se construye como un archivo de tipo librería dinámica (con extensión .dll o .so) que contiene una interfaz exportable en C/C++, es posible elaborar aplicaciones clientes utilizando diferentes lenguajes de programación o plataformas. Esto tomando en cuenta que previamente se deben construir los correspondientes envoltorios por lenguaje o plataforma para el uso de SAFET.

3.5. Búsquedas avanzadas

Debido a que en un flujo de trabajo se pueden concatenar varias consultas SQL, es posible especificar consultas avanzadas que incorporen filtros, auto-filtros, filtros recursivos y parámetros.

3.5.1. Filtros. Comos se observa en la figura 2 para cambiar de un estado se debe especificar una condición en SQL. Esto indica que en la tarea 2 se encuentra todos los registros que cumplan con la condición 1. Los registros que no cumplan con la condición 1 se encuentran almacenados en la tarea 1. Es posible concatenar esta operación y realizar filtros anidados como en la figura 3.

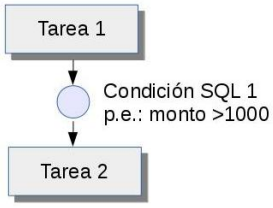

Figure 2. Generación de filtros usando SQL

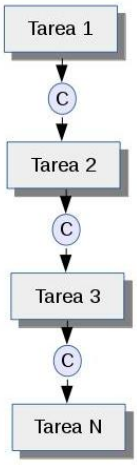

Figure 3. Generación de un auto-filtrp

Una de las ventajas de los filtros es que posible identificar rápidamente (de forma visual y clasificada) conjuntos de registros haciendo uso de todas las características del lenguaje SQL.

3.5.2. Auto-Filtros. En los flujos de trabajo es posible especificar filtros que crean de forma dinámica tareas en función de las categorías de un determinado campo. Por ejemplo, si existen un campo donde se almacena la ciudad de nacimiento de una persona (por ejemplo un estudiante), con SAFET se puede crear un conjunto de tareas que corresponden respectivamente con grupos de personas cuyas ciudades de nacimiento sean iguales. La especificación de un auto-filtro se muestra en el código 2.

```
<autofilter query="select
ciudad_nacimiento
from estudiantes"
report="yes" type="split"
tokenlink="" id="Por_tipo"
source="final" />
```

Código 2. Definición de auto-filtro

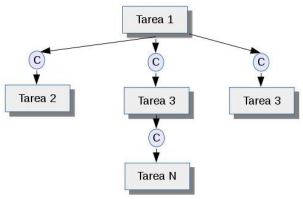

Figure 4. Generación de un filtro recursivo

El código 2 genera un conjunto de tareas en función del número de categorías que se encuentren en el campo "ciudad_nacimiento". El nombre del autofiltro se especifica en el atributo "id". Usando este nombre se puede activar o no la ejecución de esta búsqueda avanzada.

3.5.3. Filtros recursivos. Otro de los casos recurrentes en la gestión de sistema informáticos, son datos que se relacionan uno con otro y que forman una estructura de árbol. Por ejemplo, se puede citar como la dependencia entre tareas en un proyecto en un proyecto de desarrollo de software. Suponga que se desea desarrollar una aplicación y que se tienen tareas de la interfaz (frontend) y del sistema servidor (backend), a su vez, en las tareas de interfaz hay tareas para la pantalla de registros de usuario, listados, etc. y en las tareas de servidor están las correspondientes a las tareas de configuración de la base de datos, entre otros. De esta manera, se va configurando una estructura recursiva en forma de árbol. Suponga que en cada tarea existe un campo llamado "subtarea_de" que indica el padre de la tarea en cuestión, y se puede adoptar que si se encuentra en "0" no tiene tareas padres.

Usando SAFET es posible especificar fácilmente una estructura compleja utilizando un esquema similar al que se muestra en el código 2.

```
<recursivefilter filter="select
id,summarytentativedate
as info  from ticket
where depend_id={{id}}"
initial="{#ById}"
report="yes" type="split"
tokenlink="" id="por_sub"
source="final" />
```

Código 3. Definición de filtro recursivo

El código 3 genera una estructura similar a la que se muestra en la figura 4. De esta manera, se genera automáticamente el árbol visualizando la dependencia de tareas.

3.5.4. Parámetros. Una de las funcionalidades estándares de los sistemas informáticos, es permitir que el usuario realice búsquedas parametrizadas, esto es, que puede a través de una interfaz de usuario especificar parámetros que restrigan o amplian el conjunto de registros a seleccionar.

En los flujos de trabajo es posible especificar parámetros que pueden se actualizan la generación de datos, afectando

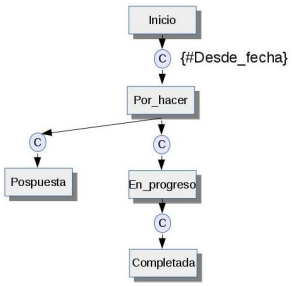

Figure 5. Ejemplo del uso de parámetros en un flujo de trabajo.

de forma concatenada y organizado los filtros, auto-filtros y filtros recursivos.

```
  <connection
  source="Hasta_fecha"
  query="select tentativedate
  from ticket"
  options="&lt;{#Porfecha}" >
</connection>
```

Código 4. Especificación de un parámetro

El código 4 muestra la especificación de un parámetro "#{Desde_fecha} "en un la conexión de una tarea con otra en un determinado flujo de trabajo. En este ejemplo se muestran tareas que se encuentren agendadas posterior a una fecha indicado por el usuario. En este flujo existen conjuntos de registros clasificados por su Estado: Por_hacer, En_progreso, Pospuestos, Completados. Es importante señalar que al indicar el parámetro se ven afectados todos los conjuntos de registros agrupados en esta tarea según se muestra en la figura 5, esto es, en la tarea "Completadas" se encuentran todas las tareas del proyecto que estén completadas, y así para cada una de ellas.

3.6. Visualización

La generación de informes, listados y reportes son funcionalidades estándares en cualquier sistema de información. En este sentido, SAFET genera salida de datos utilizando el formato JSON, de tal manera que pueda ser leída fácilmente por cualquier plataforma o lenguaje de programación. Los tipos de salida en SAFET son dos: reportes textuales y gráficos de flujos. El primero de ellos se generan a partir de los conjuntos de datos que se encuentran en cada tarea y que se identifican como variables (Variable). El segundo tipo, genera datos que pueden ser texturizados (en nuestro caso se usa graphviz) para obtener una visualización acorde con lo especificado en un determinado flujo de trabajo.

Por ejemplo, para obtener un reporte de los tareas por hacer correspondiente al sistema de gestión de tareas de proyectos de software (Ver figura 5) descrito anteriormente, se debe especificar la variable correspondiente a la tarea Por_hacer . En este caso se obtendrá un reporte en formato JSON con todas las tareas correspondiente a este conjunto de datos. Esta acción es aplicable para cualquier tarea, ya sea

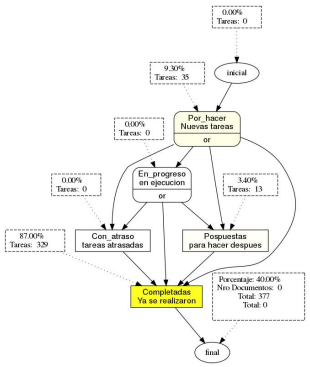

Figure 6. Flujo de trabajo de tareas de proyectos texturizado con Graphviz.

esta se genera afectada por un Autofiltro, un filtro recursivo o parámetros.

Para el segundo tipo de visualización, se obtiene un gráfico del flujo de tarea, que dibuja un conjunto de estadística, tal como se muestra en la figura 6

4. Casos de ejemplo

Para demostrar el uso de SAFET en el desarrollo rápido de aplicaciones, que puedan integrarse con otros sistemas y articularse con ambientes cliente-servidor, dispositivos móviles y lógicas complejas, seguidamente se describen dos casos que ilustran los aportes esenciales de la propuesta.

4.1. Lista de Tareas Móvil

Este aplicación se deriva del caso de Lista de Tareas estándar *ToDoList*). En este caso, se cuenta con una arquitectura basada en servicios. SAFET se utiliza para implementar los servicios que consume una aplicación cliente implementada para dispositivos Android. Una captura de pantalla del cliente Android se muestra en la figura 7. Para el registro e inicio de sesión de los usuarios se utiliza el módulo de Autenticación y Autorización de SAFET. Esto permite la adición remota de usuarios y el control de autorización de las operaciones en el servidor.

Las operaciones básicas se listan en la tabla 4. Existen dos categorías: la gestión de tareas (tickets) y la gestión de proyectos. En la gestión de tareas se utiliza el esquema CRUD+, ya que a las acciones del CRUD estándar se agrega el cambio de estado de la tarea (Por_hacer, En_progreso, Completado, Con_atraso y pospuestas). Para esta acción corresponden los estados que se muestran en el gráfico de la figura 5.

Con la aplicación cliente es posible realizar búsquedas avanzadas filtrando tareas por fecha, tipo o proyecto, de forma que se pueda obtener un resultado de cualquiera de las combinaciones de filtro, utilizando la característica de parámetros de SAFET.

En esta aplicación también se aplican los auto-filtros, de manera que se puedan obtener conjuntos de tareas agrupadas

Figure 7. DoItAll. Aplicación Cliente consumiendo servicios SAFET

Table 4. ACCIONES SAFET

Acciones	
CRUD+ ticket	Agregar,Modificar,Eliminar,Siguiente _estado
CRUD proyectos	Agregar,Modificar,Eliminar,Ver

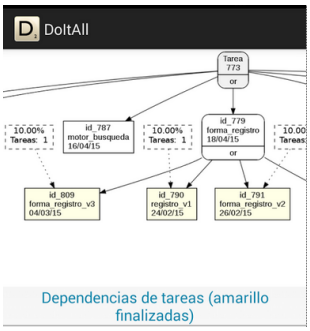

Figure 8. DoItAll. Aplicación del filtro recursivo

por proyecto, por tipo, obteniendo sus respectivo porcentajes y número de valores.

En este caso el filtro recursivo se aplica para mostrar la dependencia de tareas. Se tiene un campo de “dependiente_de” que se asocia con una tarea Padre, y así sucesivamente. Se puede obtener una gráfica de dependencia similar a la mostrada en la figura 8.

4.2. Comercio electrónico

Las aplicaciones de comercio electrónico, es decir, la venta y compra de objetos, artículos o servicios a través de internet utilizando generalmente un portal Web son de uso común hoy en día y plantean diversos retos que tienen que ver con funcionalidades específicas en los temas de búsqueda y gestión de datos.

En este sentido, se ha implementado un portal Web utilizando SAFET, y se ha configurado sus módulos de la siguiente forma:

Figure 9. Uso de auto-filtros en aplicación de comercio electrónico

Figure 10. Búsqueda de producto generada por SAFET

Autenticación y Autorización: se ha configurado el registro de usuarios utilizando el correo electrónico para verificar la identidad. Se habilitó el cambio de contraseña, la recuperación de contraseña y la gestión del perfil de usuario.

Analizador de datos: este componente permite para implementar los formularios para el ingreso de productos, estableciendo todas las tareas de gestión adición, modificación, vista y eliminación.

Motor de flujo de trabajo: se utiliza para seguir el estado de los productos dentro del portal o tienda en internet. Los estados posibles son: Borrador, Publicado, Vendido y No_activo.

Búsquedas y visualización: se implementó un conjunto de auto-filtros, de tal manera de que en el momento que el usuario realice la búsqueda de un producto se generen las categorías de forma dinámica en que se encuentra el producto o tipo de producto a buscar. Por ejemplo, si el usuario busca "computadoras", aparecerán un conjunto de datos generados por autofiltros, en función de su marca, precio, y ciudad del vendedor, entre otras características, tal como se muestra en la figura 9.

Uno de los datos que se obtiene de los auto-filtros es el número de elementos (productos) que existen en cada categoría generada. Al pulsar sobre el la categoría se muestran en la lista del lado derecho (Ver figura 9) los elementos correspondientes al conjunto de datos del filtro aplicado.

5. Conclusiones

En este trabajo se desarrolló una propuesta para la gestión avanzada de datos, utilizando un modelo que parte del uso de flujos de trabajo y que trabaja de forma coherente con el lenguaje SQL y el análisis de información textual.

El módulo de gestión de Autenticación y Autorización permite desvincular la gestión de accesos y permisos sobre los activos de información, de los datos específicos y centrales del sistema informático, colocando esta gestión en una área separada, de tal forma que puede aumentarse los niveles de seguridad, producto del aislamiento y la visión basada en componentes.

La adición de tres conceptos como lo son: Los parámetros, los auto-filtros y los filtros recursivos, logran modelar situaciones complejas que conmumente se presentan en los sistemas informáticos de hoy en día, y que su especificación y programación consumen cuantiosos recursos en las fases de construcción, pruebas y despliegue.

SAFET puede desplegarse en diferentes tipos de arquitecturas y plataformas, ya que se inserta en el modelo de computación en la nube basada en servicios [13]. La visión descrita durante el desarrollo del trabajo es la orientación de los procesos de entrada y salida de los sistemas informáticos al uso de un lenguaje sencillo y estándar que mejore los procesos de migración e integración.

Los casos de ejemplo descritos muestran características distintivas de SAFET con respecto a otros modelos o plataformas de gestión de datos. Se subraya la importancia de orientar el desarrollo a tareas de especificación de los modelos de datos y construcción de reglas basadas en flujos de trabajo, que redunde en el rápido proceso de construcción robusta de aplicaciones.

References

[1] Hitpass, Bernhard. Business Process Management (BPM): Fundamentos y Conceptos de Implementacion: Fundamentos y Conceptos de Implementacion. BHH Ltda. Santiago de Chile, Chile. 2012.

[2] Grimbol, Justin. The CRUD Masters. EraserHead Press. Portland, USA.2011.

[3] Flanagan, D. (2011). Javascript: The definitive guide. O'Really. CA, USA.

[4] De Miguel, A., Piattini, M., Marcos, E. Diseño de Base de Datos relacionales. Alfaomega, Colombia (2000)

[5] Matjaz Juric and Benny Mathew. Business Process Execution for Web Services BPEL and BPEL4WS. 2 Ed. Packt Publishing.. 2006.

[6] Elmasri, R. and Navathe, S. Fundamentals of Database Systems. Sixth Edition. Addison-Wesley. 2011.

[7] Ray, Erik. Learning XML: Creating Self-Describing Data. 2nd Edition. O'Reilly. USA. 2003.

[8] W3C. Guide to the w3c xml specification (xml spec) dtd. 2.1. Disponible: http://www.w3c.org/XML/1998/06/xmlspec-report-v21.htm (1998)

[9] Stallings, W. (2003). Fundamentos de Seguridad en Redes. Aplicaciones y estándares. Madrid España.: Pearson.

[10] Grune, D., Bal, H. Jacobs, C., Langendoen, K. Diseño de Compiladores Modernos. Mc Graw Hill. Madrid, España (2007)

[11] Jensen, K. Coloured Petri Nets. Basic Concepts, analysis methods and practical use. EATCS monographs on Theorical Computer Science.Springer-Verlag, Berlin (1996)

[12] (2016). SAFET: Sistema Automatizado de Firma Electrónica. Available on https://github.com/tibisay/pysafet/tree/bootstrap

[13] Sharma, S. (2016). Evolution of as-a-Service Era in Cloud. Available on arxiv.org: http://arxiv.org/pdf/1507.00939.pdf

[14] Bravo, V. Araujo A., SAFET: Sistema para la generación de aplicaciones de firma electrónica. Revista Puente. Vol 6. Número 1. Bucaramanga, Colombia. 2011.

[15] Campderrich Falgueras, Benet. Ingeniería del Software. Editorial UOC. Barcelona, España. 2003.

[16] Sistema Automatizado de Firma y Estampillado Electrónico de tiempo (SAFET). CENDITEL. Disponible: http://seguridad.cenditel.gob.ve/safet (2009)

[17] DTD SAFET Workflow Specification. Disponible en: https://github.com/tibisay/pysafet/blob/master/pysafet/websafet/rc/yawlworkflow.dtd

[18] DTD SAFET Input Specification. Disponible en: https://github.com/tibisay/pysafet/blob/master/pysafet/websafet/rc/yawlinput.dtd